User Implementation Manual

App class

The app class contains the start() method, the loadFXML() method, and the main() method.

-The start() method loads the window for the GUI, and calls the loadFXML() method.

```
@Override
public void start(Stage stage) throws IOException {
    scene = new Scene(loadFXML(fxml:"primary"), width:750, height:750);
    stage.setScene(scene);
    stage.show();
}
```

-The loadFXML() method loads the primary.fxml file, which sets up the rest of the GUI, being the start button, labels, and further on the board grid.

```
private static Parent loadFXML(String fxml) throws IOException {
    FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource(fxml + ".fxml"));
    return fxmlLoader.load();
}
```

-The main() method launches the program that starts everything when run.

```
Run | Debug | Run main | Debug main
public static void main(String[] args) {
    launch();
}
```

Controller class

The controller class contains the logical aspects of the code.

-The foodSpawn() method randomly selects a spot on the grid for it to spawn, it also calls the foodOverSnake() method.
-The foodOverSnake() method checks if the food spawned on the same square of the snake and generates another spot for it until it finds an empty square.

```
private boolean foodOverSnake(int row, int column) {
    for (Pane snakeSegment : snake.getSnake()) {
        int snakeRow = GridPane.getRowIndex(snakeSegment);
        int snakeColumn = GridPane.getColumnIndex(snakeSegment);
        if (snakeRow == row && snakeColumn == column) {
            return true;
        }
    }
    return false;
}
```

-The Controller() method sets up the grid for the game board.

```
public Controller() {
    paneList = new ArrayList<>();
    paneArray = new Pane[15][15];
}
```

-The initGame() method is what initializes the board, and creates the checkerboard pattern
-The handleStartClick() method is initiated when the start button is pressed, and begins the game. It calls the initGame() method and the startGame() method, it also hides the start button again until the game is over.

```
@FXML
public void handleStartClick() {
    startButton.setVisible(value:false);
    gameOverLabel.setVisible(value:false);
    initGame();
    startGame();
    gameBoardGrid.requestFocus();
}
```

-The startGame() method begins the game. It moves the snake, changes its direction, checks if it runs into a wall(and ends the game), checks if it runs into itself(ends the game), checks if the

snake ate the food and if it does it updates the score and makes it grow, and after every 10 it increases the speed of the snake(decreases the delay between each movement).
-The move() method moves the direction of the snake if it changes.

```
@FXML
void move(KeyEvent e) {
    if (e.getCode().equals(KeyCode.UP) && currentDir != KeyCode.DOWN) {
        changeDir = KeyCode.UP;
    } else if (e.getCode().equals(KeyCode.DOWN) && currentDir != KeyCode.UP) {
        changeDir = KeyCode.DOWN;
    } else if (e.getCode().equals(KeyCode.LEFT) && currentDir != KeyCode.RIGHT) {
        changeDir = KeyCode.LEFT;
    } else if (e.getCode().equals(KeyCode.RIGHT) && currentDir != KeyCode.LEFT) {
        changeDir = KeyCode.RIGHT;
    }
}
```

-The gameOver() method ends the game and is called if the snake runs into the wall or its own body.

```
private void gameOver() {
    gameOn = false;
    if (score == 222) {
        gameOverLabel.setText(value:"YOU WIN!!!!");
        gameOverLabel.setVisible(value:true);
        startButton.setVisible(value:true);
    } else {
        gameOverLabel.setAlignment(Pos.CENTER);
        gameOverLabel.setVisible(value:true);
        startButton.setVisible(value:true);
    }
}
```

-The onKeyPressed() method checks if a key is pressed and calls the move() method to change its direction.

```
@FXML
void onKeyPressed(KeyEvent e) {
    move(e);
}
```

-The gameStep() method continues the game and keeps checking everything until the game is over.
-The faceRotation() method rotates the smiley face that indicates the direction the snake is moving.

```
private double faceRotation(KeyCode direction) {
    switch(direction) {
        case UP:
            return 270;
        case DOWN:
            return 90;
        case LEFT:
            return 180;
        case RIGHT:
            return 0;
        default:
        return 0;
    }
}
```

-The snakeDirection() method changes the position of the face when it rotates, centering it on the tile it is currently occupying.

```
private void snakeDirection(Pane pane, KeyCode direction) {
    double faceX = pane.getWidth() / 2;
    double faceY = pane.getHeight() / 2;
    face.setLayoutX(faceX * 2 -15);
    face.setLayoutY(faceY - 20);
    face.setFont(javafx.scene.text.Font.font(size:25));
    face.setStyle(value:"-fx-text-fill: white;");
    face.setRotate(faceRotation(direction));
    pane.getChildren().clear();
    pane.getChildren().add(face);
    if (faceRotation(direction) == 0) {
        face.setLayoutX(faceX * 2 -15);
        face.setLayoutY(faceY - 20);
    } else if (faceRotation(direction) == 90) {
        face.setLayoutX(faceX-5);
        face.setLayoutY(faceY - 5);
    } else if (faceRotation(direction) == 180) {
        face.setLayoutX(value:0);
        face.setLayoutY(faceY - 15);
    } else if (faceRotation(direction) == 270) {
        face.setLayoutX(faceX-10);
        face.setLayoutY(- 10);
    }
}
```

Snake class

The Snake class contains the snake() method, getSnake(), head(), contains(), grow(), and move() methods.

-The snake() method starts the snake off with the length of 3 and makes it the color blue. It is stored in a 2 dimensional array.

```
public Snake(Pane[][] paneArray) {
    this.paneArray = paneArray;
    this.snake = new LinkedList<>();
    for (int i = 0; i < 3; i++) {
        Pane pane = paneArray[7] [i + 3];
        pane.setStyle(value:"-fx-background-color: blue;");
        snake.add(pane);
    }
}
```

-The getSnake() method returns the placement of the snake.

```
public LinkedList<Pane> getSnake() {
    return snake;
}
```

-The head() method returns the placement of the head of the snake.

```
public Pane head() {
    return snake.getLast();
}
```

-The contains() method returns if the parameter is a part of the snake.

```
public boolean contains(Pane pane) {
    return snake.contains(pane);
}
```

-The grow() method adds another blue tile to the snake's body when it eats the food.

```
public void grow(Pane next) {
    next.setStyle(value:"-fx-background-color: blue;");
    snake.add(next);
}
```

-The move() method removes the tail when the snake moves forward, and changes the previous tile's color to the correct shade of green.

```java
public void move(Pane next) {
    grow(next);
    Pane tail = snake.removeFirst();
    int row = GridPane.getRowIndex(tail);
    int column = GridPane.getColumnIndex(tail);
    if ((row + column) % 2 == 0) {
        tail.setStyle(value:"-fx-background-color: limegreen;");
    } else {
        tail.setStyle(value:"-fx-background-color: lightgreen;");
    }
}
```